Even Semester (2021)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Group Work)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **James**  **Ivan**  **David** | | **Student ID Number**  **2101695176**  **2101693920**  **2101693933** | |
| 1.  2.  3. | | **Barlian**  **Suratno**  **Honasan** |  | |
|  |  |
| **Course Code** | **: COMP6048** |  |  | | **Course Name** | | **: Data Structures** | |
| **Class** | **: L2AC** |  |  | | **Name of Lecturer(s)** | | **:** 1. Tri Budiono | |
|  |  |  |  | |  | | 2. Raymond Kosala | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment** | : **Map Navigation** | |  |  | |  | |  | |
| **Type of Assignment**  **Submission Pattern**s | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 5-30-2018** |  |  | | **Submission Date** | | **: 5-30-2018** | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. James Barlian
2. Ivan Ezechial Suratno
3. David Honasan

**“Map Navigation”**[**https://github.com/ezeutno/DataStructure2018\_Final-Project**](https://github.com/ezeutno/DataStructure2018_Final-Project)

**Name : James Barlian 2101695176**

**Ivan Ezechial Suratno 2101693920**

**David Honasan 2101693933**

# Problem Description

To Find the shortest path in a map created by the users.

# Alternative Data structure to solve the problem

In this Program we used vector and linked list.

**Function of the program:** This program is used for finding the shortest path in a map when given two points. The program will give you all other possible routes of getting to the destination. Basically we are creating a navigational application that will give you the best route to your destination as well as backup routes.

# Design/Analysis

Text

Map Text read and store

Navigation

Graphs

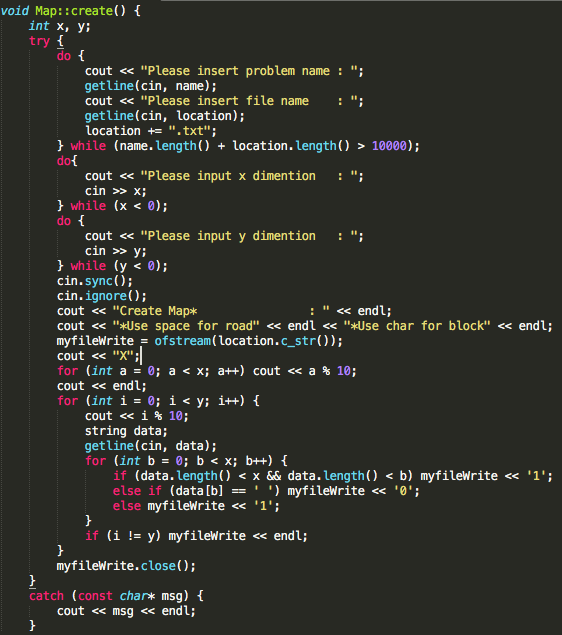
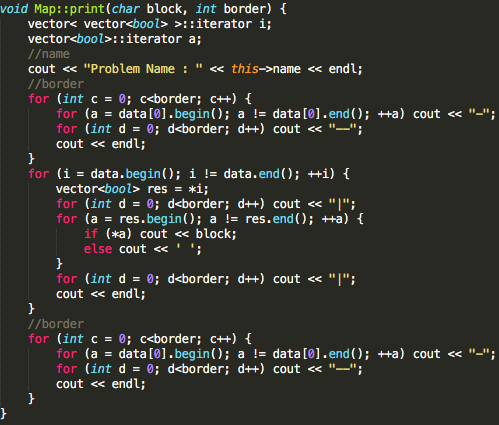
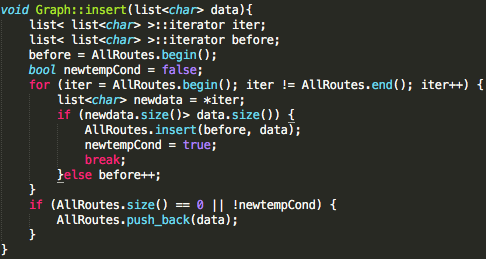
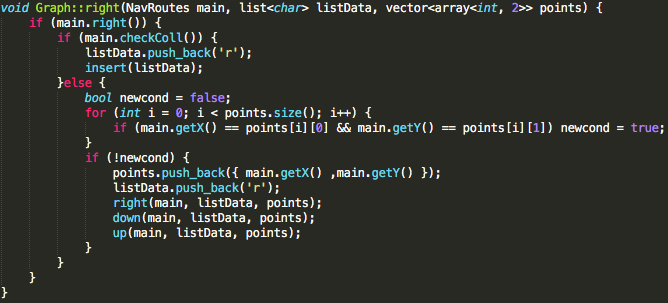
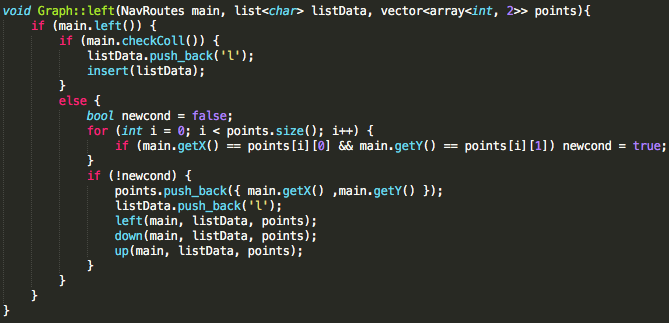
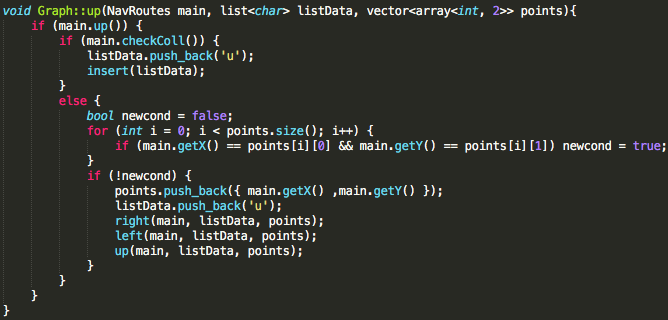
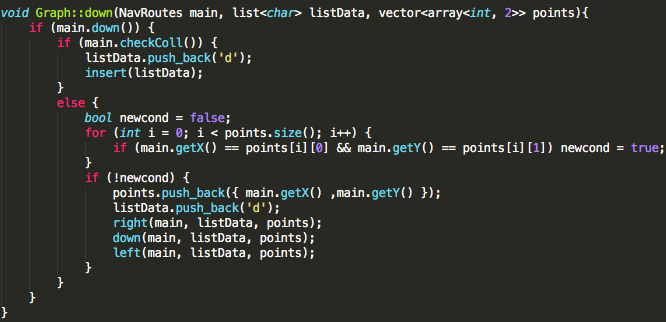
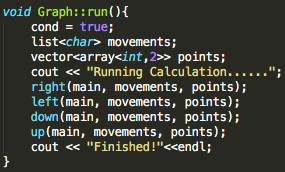
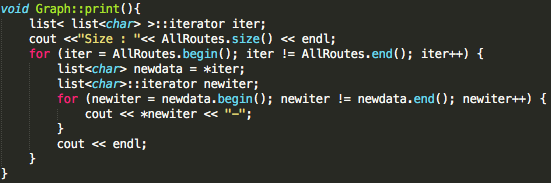
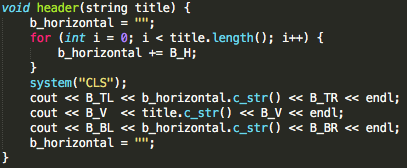
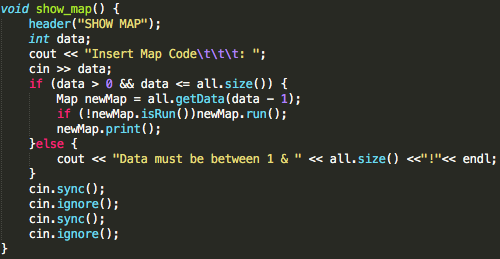
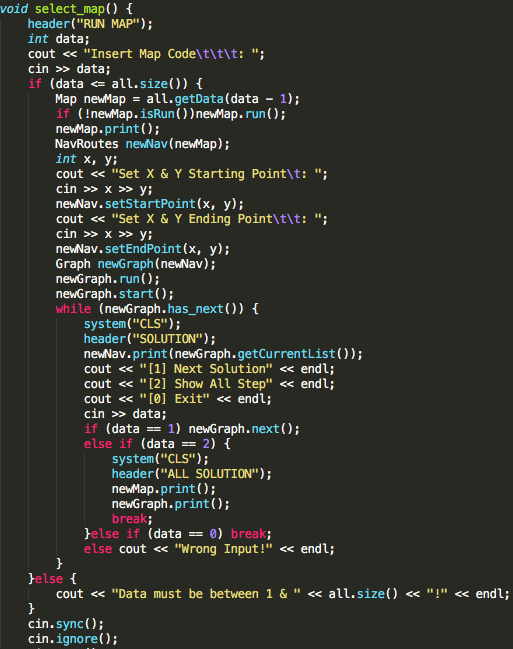
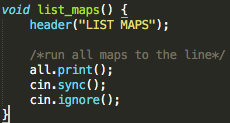
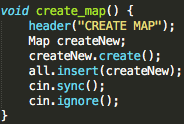
Sorted Linked List

All Possible Routes Found

# Explanations of Functions

1. **( Nav.h ) Navigation**
   * Declaration of class members functions, constructor, and attributes:
   * Private:
     1. Map data;
     2. int size[2];
     3. int startPoint[2];
     4. int currPoint[2];
     5. int endPoint[2];
   * Protected:
     1. void create();
   * Public:
     1. NavRoutes();
     2. NavRoutes(Map & data, int start[], int end[]);
     3. void setStartPoint(int x, int y);
     4. void setEndPoint(int x, int y);
     5. int getX();
     6. int getY();
     7. bool checkColl();
     8. bool right();
     9. bool left();
     10. bool up();
     11. bool down();
2. **( Nav.cpp ) Navigation**
   * In the Navigation class we can set the start and end points by using the setStartPoint() and setEndPoint() Methods.
   * checkColl() will check if there are collision in all of the directions before making choosing the next path.
   * right(), left(), up(), down() are the only directions that are allowed for this navigational application.
3. **( mapText.h )**
   * Declaration of class members functions, constructor, and attributes:
   * Private:
     1. string name;
     2. string location;
     3. bool runCond;
     4. ofstream myfileWrite;
     5. ifstream myfileRead;
     6. vector< vector<bool> > data;
   * Public:
     1. Map();
     2. Map(string loc, string = "Problem");
     3. Map(const Map& data);
     4. string getName();
     5. bool isRun();
     6. void run();
     7. bool getData(int x, int y);
     8. void setName(string name);
     9. void setLocation(string loc);
     10. int getSize(char position);
     11. string getLocation();
     12. void create();
     13. void print(char = '\*', int = 1);
     14. void operator=(Map & a);
     15. friend ofstream & operator<<(ofstream &, Map & b);
     16. friend ostream & operator<<(ostream &, Map & b);



1. **( mapText.cpp )**
   * In the Map class the user is able to create their own map using the void create() Method.
     1. When creating the map the program will ask the user to insert a problem name, a file name, the x dimension, and y dimension.
     2. Then after it will tell the user to enter specific characters to build the walls and an empty space for the road.
     3. 
   * After creating a map of their choosing, then the void print() method can be used to show the map to the user.
   * 
   * Void Run() will retrieve the map from a .txt file and insert it into a vector.
   * Void setName () will set the name of the map.
   * Void getName() will return the name of the map.
2. **( graph.h )**
   * Declaration of class members functions, constructor, and attributes:
   * Private:
     1. NavRoutes main;
     2. bool cond;
     3. linkedlistADT< linkedlistADT<char> > AllRoutes;
     4. void insert(linkedlistADT<char> data);
   * Protected:
     1. void right(NavRoutes main, linkedlistADT<char> list, vector< array<int,2> > points);
     2. void left(NavRoutes main, linkedlistADT<char> list, vector< array<int, 2> > points);
     3. void up(NavRoutes main, linkedlistADT<char> list, vector< array<int, 2> > points);
     4. void down(NavRoutes main, linkedlistADT<char> list, vector< array<int, 2> > points);
   * Public:
     1. Graph();
     2. Graph(NavRoutes main);
     3. bool isRun();
     4. void run();
     5. void print();
3. **( graph.cpp )**
   * The Void insert() function will insert the linked list into a linked list based on the size current of the linked list.
   * 
   * Void right() will move the current position by 1 to the right and the next pathing option cannot consist of going back to the left.
     1. It will also push\_back ‘ r ‘ into the list.
   * 
   * Void left() will move the current position by 1 to the left and the next pathing option cannot consist of going back to the right.
     1. It will also push\_back ‘ l ‘ into the list.
   * 
   * Void up() will move the current position by 1 upwards and the next pathing option cannot consist of going back down.
     1. It will also push\_back ‘ u ‘ into the list.
   * 
   * Void down() will move the current position by 1 downwards and the next pathing option cannon consist of going back up.
     1. It will also push\_back ‘ d ‘ into the list.
   * 
   * Void run() will move the position from its starting point to the end point using nav.h methods until it reaches its destination.
     1. 
   * Void print() will print all the possible routes that it takes from the starting point to the end point of the map.
   * 
4. **( Main.cpp )**
   * Outside the main we have a series of function that deals with out interface.
   * First we have void header(string title) which simply just puts a header in the interface menu.
     1. 
   * Second, we have void show\_map() and when a user inputs a map code the selected map will then be shown for the user. The inputted number must not exceed the amount of maps that is stored in the vector.
     1. 
   * Third, we have void select\_map() which is going to select whichever map the user inputs, then it will ask for a set of (X,Y) coordinates to be inputted as a starting point as well as an end point. After doing so the first solution will be shown to the user, followed by an option to see alternatives solutions or all possible solutions listed.
   * 
   * Fourth, we have void list\_maps() which prints out a list of all the maps that are stored in the vector.
     1. 
   * Lastly, we have void create\_map() which allows the user to create a map. (See mapText.cpp for void create())
     1. 
   * Inside the main we have a list of options to choose from when starting the application.
     1. [1] Run map
     2. [2] List all maps
     3. [3] Create new map
     4. [4] Show map
     5. [0] Exit
   * Using switch to set up different cases for each option.
   * 